Software Design Document

Victorian Accident Analysis Project

Harris Huntington, Eric Martin

Table of Contents

[1.0 System Vision 3](#_Toc113217436)

[1.1 Problem Background 3](#_Toc113217437)

[1.2 System Overview 3](#_Toc113217438)

[1.3 Potential Benefits 3](#_Toc113217439)

[2.0 Requirements 4](#_Toc113217440)

[2.1 User Requirements 4](#_Toc113217441)

[2.2 Software Requirements 4](#_Toc113217442)

[2.3 Use Cases & Use Case Diagrams 5](#_Toc113217443)

[3.0 Software Design and System Components 6](#_Toc113217444)

[3.1 Software Design 6](#_Toc113217445)

[3.2 System Components 7](#_Toc113217446)

[3.2.1 Functions 7](#_Toc113217447)

[3.2.2 Data Structures / Data Sources 8](#_Toc113217448)

[3.2.3 Detailed Design 8](#_Toc113217449)

[4.0 User Interface Design 9](#_Toc113217450)

[4.1 Structural Design 9](#_Toc113217451)

[4.2 Visual Design 9](#_Toc113217452)

# System Vision

## Problem Background

This Victorian Accident Analysis dataset is a set of data containing road crash statistics from the five years between 2015 and 2020 within the Australian state of Victoria. This team has been tasked to develop a program which will allow users to analyse certain data and trends from the dataset.

## System Overview

This software will allow a user to analyse data from the dataset through a GUI. The user will be prompted to select a time frame (days) and when selected, information about all the accidents within that time frame will be displayed.

The average number of accidents that occur in this time frame can be viewed as a graph. To narrow down accident logs, a text input will allow a user to enter key words such as pedestrian, truck, car etc. Any accidents that contain these keywords will be shown to the user.

An alcohol use analysis feature will allow users to see a visual representation of alcohol impact. This will include accidents with alcohol related causes.

Another feature will be an analysis of the most common average time accidents occur on each day of the week.

## Potential Benefits

This will have several benefits to a few different user groups. Governments will be able to use this data to understand what is causing the most accidents, and work on changes that can help reduce these accidents thus saving damages and lives. As well as insurance companies can use this data to help model their plans and get a better idea of when, where, and why accidents happen and to be able to offer more suitable plans for different groups of people.

This will also provide important information to general road users and allow them to understand how accidents happen on the road and better prepare themselves and change their habits to help reduce the number of accidents they could cause or be a part of.

# Requirements

## User Requirements

The user needs to be able do the following:

* Select use (see accidents, view accident graph, open alcohol use section)
* Input a date timeframe
* Scroll down through multiple accident logs
* Open and close graphs
* Input text
* Bookmark certain accident logs

The users of this software are likely to be the government and insurance companies, who need to use accident data to create policies.

## Software Requirements

The software will require the following:

1. A data set will be stored within the program
2. The program shall accept 2 date inputs
3. If an object in the data set has a date between the two accepted dates, then it will be displayed.
4. The program shall accept up to 2 keywords through a text input.
5. If an object in the dataset contains these key words in its Accident\_Type attribute, then it will be displayed
6. The program shall accept the date and text inputs at the same time when necessary.
7. The program shall modify the graph shown with respect to the documents being displayed at the time.
8. The program will provide trends in regards to alcohol consumption and accidents.

## Use Cases & Use Case Diagrams

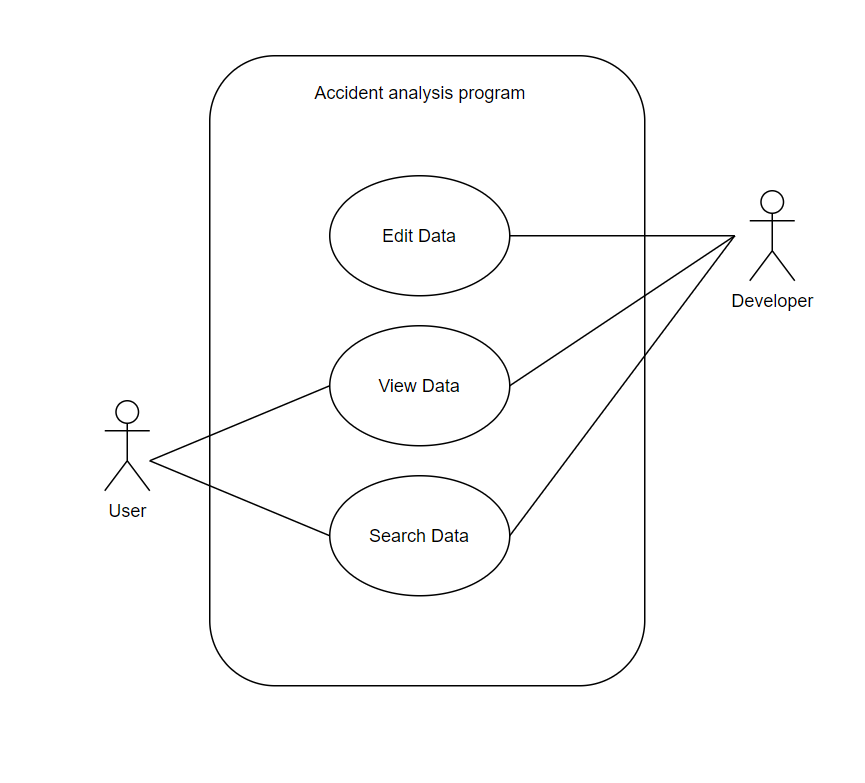
|  |  |
| --- | --- |
| Use Case ID | 1 |
| Use Case Title | Search by date |
| Primary Actor | User |
| Trigger | User inputs dates and clicks search |
| Success Scenario | The data matching the time frame is loaded and displayed |
| Failure Scenario | No data matching the time frame was found |

|  |  |
| --- | --- |
| Use Case ID | 2 |
| Use Case Title | Hourly Data |
| Primary Actor | User |
| Trigger | User inputs date, selects the hourly filter, and clicks search |
| Success Scenario | The data matching the time frame is loaded and displayed as accidents every hour |
| Failure Scenario | No data matching the time frame was found |

|  |  |
| --- | --- |
| Use Case ID | 3 |
| Use Case Title | Sort by keyword |
| Primary Actor | User |
| Trigger | User selects type filter and types a keyword into the input box, then clicks search |
| Success Scenario | Data with matching keywords is loaded and displayed on the graph. |
| Failure Scenario | No data matching the keyword given is found in the dataset |

|  |  |
| --- | --- |
| Use Case ID | 4 |
| Use Case Title | Alcohol impact |
| Primary Actor | User |
| Trigger | User selects the alcohol impact tab, selects trends, and clicks search. |
| Success Scenario | The alcoholic impact on accident trends are displayed. |
| Failure Scenario | No data is found |

Diagram

Description automatically generated

# Software Design and System Components

## Software Design

Diagram

Description automatically generated

## System Components

### Functions

1. loadData()
   1. The loadData function will use user inputs to check which pieces of data from the dataset need to be loaded.
   2. Input parameters
      1. Day1: string, used with Day2 to define a time period within the dataset (start)
      2. Day2: string, used with Day1 to define a time period within the dataset (end)
      3. Filter: string (All, hourly, or Type). Used to filter how much data is loaded.
      4. Keyword: string. Used in the checkKeywords function to match data.
   3. Side effects cause by this function will include changing global variables.
   4. The function will return a list of data from that dataset that matches the inputs (date, filters, keywords) that a user decides on.
2. checkKeywords()
   1. checkKeywords() will be used to check each bit of data so see if the accident\_type attribute contains the keyword the user inputted.
   2. Input Parameters
      1. Word: string. Used as the word that the function checks the dataset against.
   3. Side effects should not be present when running this function.
   4. checkKeywords() will return a list of objectIDs for use in the loadData function.
3. displayGraph()
   1. This function displays the data from the loadData function on a graph for the user.
   2. Input Parameters
      1. Count: int. Number of data entries to be displayed
      2. isAlc: bool. Determines whether alcohol was a cause of the accident
      3. start: string. The start date to use as point 0 on the graph (x-axis)
      4. end: string. The end date to use as the last point on the graph(x-axis)
   3. Side effects of this function could include changing global variables for the graph scaling, depending on the amount of data being displayed
   4. The function will return the number of accidents each day (or hour if filtered hourly). The graph’s y axis will represent this number (highest y value being 10 above the max number returned by the function). The function will draw a line graph to show trends.

### Data Structures / Data Sources

1. Graph Data. The values given by the displayGraph function will be stored in an array of integers, which will be used to draw the graph. [0] will be the first
2. Accident Data. Each accident will be stored as an object with a Class ‘Accident’.

|  |  |
| --- | --- |
| Class Name | Accident |
| Class Attributes | accidentID: String  Date: Date  isAlc: Boolean  time: Integer  type: String  detail: String |

This class will be used by the loadData function and the checkKeywords function.

### Detailed Design

Pseudocode for all non-standard / non-trivial algorithms that operate on data structures

# User Interface Design

This user interface design used draw.io to create the structural design. With this the navigational and information structure of the product was able to be devised. A flowchart of sorts was used to demonstrate this design, allowing for a clear and easy path to be formed to dictate the flow of the structure of the information and navigation of the product.

Figma was then used to create a visual wireframe of the product. This allowed for a clear way to template the final product, primarily focusing on the physical layout of all the elements. The rest of the design elements, fonts, colours, etc, are then discussed to provide a fleshed-out plan for the visual design of the product.

## Structural Design

Diagram

Description automatically generated

The structural design of this product is a single page product with everything except the data display always on the screen. This approach was taken to allow the user to clearly see and find anything they are looking for at a glance without digging through menus or flipping through pages. This was done by splitting the page vertically into a data display side on the right-hand side, and the analysis options on the left-hand side. The right-hand side containing the outputted data, is non interactable by the user, instead is only there to display the requested data. The left-hand side of the product however is where all the menus are located. These are designed to be all displayed at one time, with the user selecting their options, and then confirming with the search button. This was done to make it as easy as possible for the user to understand what options they have, and how they can interact with the displayed options to output their desired analysis data.

## Visual Design

Diagram

Description automatically generated

As seen in this mock-up wireframe, the page visually follows the navigational design explained in the previous section. The data or graph to be analysed is displayed on the right-hand side, while the left-hand side contains all the options to select what is being analysed. As there are two main options for analysis, selecting by dates, and by alcohol impacts, those two options are given at the top of the left-hand side. When one of these options is selected, it will grey out the option box for the other option. All the presented options are displayed in a clear easy to understand design, allowing the user to follow the page from top to bottom to make their selection, narrow it down, and then confirm the search.

The font for this project will be the standard Calibri, as this is easy to read and instantly understood which will blend right in with the general visual nature of the project to not be overly obtrusive to the user. No icons will be used within this project for that reason, as well as to reduce clutter as they are not needed in a project such as this.

This project will use mostly muted and neutral colours, primarily focusing on blues and greys. This will allow for the visuals of the project to not take away from the use of project by distracting the user. This colour scheme also ensures that the visuals do not contrast against the sobering reality that this data is dealing with the accidents, injuries, and deaths of real people.